Careful abstraction from instance families in memory-based language learning

van den Bosch, A.

Published in:
Journal of Experimental and Theoretical Artificial Intelligence

Publication date:
1999

Careful Abstraction from Instance Families in Memory-Based Language Learning

Antal van den Bosch

ILK Research Group, Computational Linguistics

Tilburg University, The Netherlands

email: Antal.vdnBosch@kub.nl

Contact: Antal van den Bosch
ILK Research Group / Computational Linguistics
Faculty of Arts
Tilburg University
P.O. Box 90153
NL-5000 LE Tilburg
The Netherlands
phone (voice) +31.13.4668260
phone (fax) +31.13.4663110

Running heading: Careful abstraction from instance families

Abstract

Empirical studies in inductive language learning point at pure memory-based learning as a successful approach to many language learning tasks, often performing better than lerning methods that abstract from the learning material. The possibility is left open, however, that limited, careful abstraction in memory-based learning may be harmless to generalisation, as long as the disjunctivity of language data is preserved. We compare three types of careful abstraction: editing, oblivious (partial) decision-tree abstraction, and generalised instances, in a single-task study. Only when combined with feature weighting, careful abstraction can equal pure memory-based learning. In a multi-task case study we find that the FAMBL algorithm, a new careful abstractor which merges families of instances, performs close to pure memory-based learning, though it equals it only on one task. On the basis of the gathered empirical results, we argue for the incorporation of the notion of *instance families*, i.e., carefully generalised instances, in memory-based language learning.

1 Introduction

Memory-based learning has been studied for some time now as an approach to learning language processing tasks, and is found by various studies to be successful, attaining adequate to excellent generalisation accuracies on realistic, complex tasks as different as hyphenation, semantic parsing, part-of-speech tagging, morphological segmentation, and word pronunciation (Daelemans and Van den Bosch, 1992a; Cardie, 1994; Cardie, 1996; Daelemans et al., 1996; Van den Bosch, 1997). Recent studies in inductive language learning (Van den Bosch, 1997; Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming) provide indications that forgetting (parts of) task instances during learning tends to hinder generalisation accuracy of the trained classifiers, especially when these instances are estimated to be exceptional. Learning algorithms that do not forget anything about the learning material, i.e., pure memory-based learning algorithms, are found to obtain the best accuracies for the tasks studied when compared to decision-tree or edited memory-based learning algorithms. The detrimental effect of forgetting exceptions is especially witnessed in grapheme-phoneme conversion and word pronunciation, although it is not fully clear yet what distinguishes these tasks from other language learning tasks such as part-of-speech tagging, base-NP chunking, or prepositional-phrase attachment.

Learning algorithms equipped with the ability to forget (e.g., editing in IB3, Aha, Kibler, and Albert (1991), or pruning in C4.5, Quinlan (1993)) do so quite strongly by default. For example, if the decision-tree learner C4.5 is not given any parameter value overrules by the user, it tends to perform a considerable amount of abstraction (i.e., decision-tree pruning) when trained on typical language learning tasks. Although this default bias tend to produce small trees which allow very fast classification, generalisation performance on some language learning tasks is markedly worse as compared to that of memory-based classifiers. However, tuning these parameters towards less pruning in decision trees or limited forgetting of instances in edited memory-based learning, i.e., *careful* (or weak) abstraction, can yield performances that are close or equal to those of pure memory-based learning, at least for some language learning tasks (Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming).

Thus, findings from recent studies leave room for the hypothesis that careful abstraction

in inductive language learning may be an equal alternative to pure memory-based learning. Advantages of the careful abstraction approach may be that, although it is not likely to outperform the generalisation accuracies obtained with pure memory-based learning easily, it may produce a more compact model of the learning material, and it may better reflect some linguistic aspects of the data. Although computational efficiency is not at the focus of this article, it can also be hypothesised that more compact representations of the learning material allow faster classification. The topic of this article is to investigate existing approaches to careful abstraction during learning, and to perform empirical tests on language learning tasks to collect indications for the efficacy of careful abstraction in inductive language learning, in comparison with pure memory-based learning.

We use the term abstraction here as denoting the *forgetting of learning material during learning*. This *material* notion of abstraction is not to be confused with the *informational* abstraction from learning material exhibited by *weighting metrics* (Salzberg, 1991; Cost and Salzberg, 1993; Wettschereck, Aha, and Mohri, 1997), and, more generally, by the abstraction bias in all memory-based learning approaches that high similarity between instances is to be preferred over lower similarity, and that only the most similar items are to be used as information source for extrapolating classifications. In this article, comparisons are made among memory-based learning algorithms that feature both material abstraction and weighting metrics. Where possible, both types of abstraction are separated to allow for the comparison of minimal pairs of algorithmic settings, and a proper discussion of the results. The empirical part of this article describes two case studies in which only parts of the total experimental matrix between algorithms, abstraction methods, metrics, and language learning tasks are covered; it is hoped that these case studies provide indications that support the broader, secondary goal of the paper: to show that apart from instances, memory-based language learning may consider *instance families*, i.e., carefully generalised instances, as working units in learning and processing.

The article is structured as follows. Section 2 summarises methods for careful abstraction in memory-based learning; it reviews existing approaches, and presents FAMBL, a new memory-based learning algorithm that abstracts carefully by merging (families of) instances in memory.

In Section 3 a range of memory-based learning algorithms performing careful abstraction is applied to the task of grapheme-phoneme conversion. While this task is represented by a relatively small data set, the second series of experiments, described in Section 4, deals with the application of FAMBL, in comparison with its parent (pure memory-based) learning algorithm IB1-IG, to a range of language learning tasks represented by large data sets of examples. In Section 5, the efficacy of careful generalisation over families of instances is discussed, and the idea of viewing these families as linguistic units is outlined. Finally, Section 5 identifies future research.

2 Careful abstraction in memory-based learning

Memory-based learning, also known as instance-based, example-based, lazy, case-based, exemplar-based, locally weighted, and analogical learning (Stanfill and Waltz, 1986; Aha, Kibler, and Albert, 1991; Salzberg, 1991; Kolodner, 1993; Aha, 1997; Atkeson, Moore, and Schaal, 1997), is a class of supervised inductive learning algorithms for learning classification tasks (Shavlik and Dietterich, 1990). Memory-based learning treats a set of labeled (pre-classified) training instances as points in a multi-dimensional feature space, and stores them as such in an *instance base* in memory (rather than performing some abstraction over them).

An instance consists of a fixed-length vector of n feature-value pairs, and information field containing the classification of that particular feature-value vector. After the instance base is built, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance*, given by a distance function $\Delta(X, Y)$ between the new instance X and the memory instance Y. The memory instances with the smallest distances are collected, and the classifications associated with these neighbours are merged and extrapolated to assign a classification to the test instance.

The most basic distance function for patterns with symbolic features is the *overlap metric* given in Equations 1 and 2; where $\Delta(X, Y)$ is the distance between patterns X and Y, represented by n features, w_i is a weight for feature i, and δ is the distance per feature.

$$\Delta(X, Y) = \sum_{i=1}^{n} w_i \ \delta(x_i, y_i) \tag{1}$$

where:

$$\delta(x_i, y_i) = 0 \ if \ x_i = y_i, \ else \ 1 \tag{2}$$

Classification in memory-based learning systems is basically performed by the k-nearest neighbour (k-NN) classifier (Cover and Hart, 1967; Devijver and Kittler, 1982), with k usually set to 1. k-NN classification coupled with equal weighting in the similarity function (e.g., for all features f, $w_f = 1$), is essentially the IB1 algorithm (Aha, Kibler, and Albert, 1991).

Early work on the k-NN classifier pointed at advantageous properties of the classifier in terms of generalisation accuracies, under certain assumptions, because of its reliance on full memory (Fix and Hodges, 1951; Cover and Hart, 1967). However, the trade-off downside of full memory is computational inefficiency of the classification process, as compared to parametric classifiers that do abstract from the learning material. Therefore, several early investigations were performed into *editing* methods: finding criteria for the removal of instances from memory (Hart, 1968; Gates, 1972) without harming classification accuracy. Other studies on editing also explored the possibilities of detecting and removing noise from the learned data, so that classification accuracy might even improve (Wilson, 1972; Devijver and Kittler, 1980).

The renewed interest in the k-NN classifier from the late 1980s onwards in the AI-subfield of machine learning (Stanfill and Waltz, 1986; Stanfill, 1987; Aha, Kibler, and Albert, 1991; Salzberg, 1991) caused several new implementations of ideas on criteria for editing, but also other approaches to abstraction in memory-based learning emerged. In this section we start with a brief overview of approaches to *editing of instances during learning*. We then discuss one approach in which memory-based learning is optimised using *oblivious (partial) decision-tree search*. We conclude our overview with a discussion of two approaches that *carefully merge instances* into more general expressions. Consequently we present FAMBL, a carefully-abstracting memory-based learning algorithm. FAMBL merges groups of very similar instances (families) into family expressions.

As a sidenote, we mention that in this section we use *grapheme-phoneme conversion* as a benchmark language learning task from which examples are drawn illustrating the functioning of the described approaches. The task is also in focus in Section 3. Grapheme-phoneme

conversion is a well-known benchmark task in machine learning (Sejnowski and Rosenberg, 1987; Stanfill and Waltz, 1986; Stanfill, 1987; Lehnert, 1987; Wolpert, 1989; Shavlik, Mooney, and Towell, 1991; Dietterich, Hild, and Bakiri, 1995). We define the task as the conversion of fixed-sized instances representing parts of words to a class representing the phoneme of the instance's middle letter. To generate the instances, windowing is used (Sejnowski and Rosenberg, 1987). Table 1 displays four example instances and their classifications. For example, the first instance in Table 1, _hearts_ (the '_' denotes empty outside-word positions), maps to class label /A:/, denoting an elongated short 'a'-sound to which the middle letter 'a' maps. In this study, we chose a fixed window width of nine letters, which offers sufficient context information for adequate performance (in terms of the upper bound on error demanded by applications in speech technology) (Van den Bosch, 1997).

Features									
1	2	3	4	5	6	7	8	9	Class
_	_	h	e	a	r	t	s	_	/A:/
_	b	o	o	k	i	n	g	_	/k/
i	t	i	e	s	_	_	_	_	/z/
_	_	_	_	b	a	s	i	c	/b/

Table 1: Example instances of the grapheme-phoneme conversion learning task. All instances are characterised by seven features (letters) and one classlabel, which is the phonemic mapping of the middle letter of the instance.

The task is deliberately picked for providing illustrations and for running the first series of experiments (cf. Section 3), because it has been claimed and demonstrated at several occasions that pure memory-based learning is successful in learning this task (Stanfill and Waltz, 1986; Wolpert, 1989), also when compared to learning methods that abstract more strongly (Van den Bosch, 1997; Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming). The task appears more sensitive to harmful effects of abstraction than other tasks investigated in the literature, and consequently, careful abstraction will be needed the most in learning grapheme-phoneme conversion. This task bias, upheld for now for the purpose of illustration, is abandoned in Section 4 when a series of experiments on a range of other language learning tasks is described.

2.1 Existing approaches

We distinguish between three types of careful abstraction during learning:

1. **Editing** (Hart, 1968; Wilson, 1972; Aha, Kibler, and Albert, 1991): removing instances according to a classification-related utility threshold they do not reach. Editing is not careful in principle, but the approaches discussed here and included in the empirical comparison (i.e., IB2 and IB3, Aha, Kibler, and Albert (1991)) collect statistical evidence for the relative harmlessness of the editing operation to be performed.
2. **Oblivious (partial) decision-tree abstraction** (Daelemans, Van den Bosch, and Weijters, 1997): compressing (parts of) instances in the instance base into (parts of) decision-trees. Part of the motivation to perform top-down induction of decision trees (TDIDT) is the presence of clear differences in the relative importance of instance features, allowing features to be strictly ordered in matching (Quinlan, 1986). The approach is dependent on the use of a feature-weighting metric.
3. **Carefully merged instances** (Salzberg, 1991; Wettschereck and Dietterich, 1995; Domingos, 1996): merging multiple instances in single generalised instances. While individual instances are usually represented by propositional conjunctions of atomic feature values, merged instances can be conjunctions of *disjunctions of* feature values, or rules with wildcards.

In the following subsections we outline approaches to each of these three types of careful abstraction during learning.

2.1.1 Editing

In memory-based learning, it seems sensible to keep any instance in memory that plays a positive role in the correct classification of other instances within memory or of new, unseen instances. Alternatively, when it plays no role at all in classification, or when it is disruptive for classification, it may be discarded from memory. These two options form the bases of two approaches to editing found in the literature:

1. *Delete instances of which the deletion does not alter the classification performance of the memory-based classifier.* Performance changes can only be measured on the instances

stored in memory. The assumption is made that lack of performance loss measured on the instances in memory, transfers to lack of performance loss on unseen instances. Early examples of this approach are the Condensed Nearest Neighbour classifier proposed by Hart (1968), the Reduced Nearest Neighbour classifier (Gates, 1972) and the Iterative Condensation Algorithm (Swonger, 1972). The IB2 algorithm (Aha, Kibler, and Albert, 1991) is a more recent example.

2. *Delete instances of which the classification is different from the majority class of their nearest neighbours.* Such instances may play a disruptive role in classification, since apparently they are positioned in a part of instance space dominated by another class than their own. Early approaches to this type of editing include Wilson (1972), Tomek (1976), and Devijver and Kittler (1980). In IB3 (Aha, Kibler, and Albert, 1991) the approach is to delete instances estimated to cause misclassifications of (unseen) neighbour instances according to a *class prediction strength* estimate. Again, such estimates can only be based on the material available in memory, but are assumed to apply to unseen material as well.

Aha, Kibler, and Albert (1991) describe a class of instance-based (memory-based) learning algorithms that learn incrementally, i.e., instance by instance. Two of these algorithms, IB2 and IB3, are compared in the case study of Section 3. We describe them briefly here. First, IB2 edits instances from memory that are classified correctly by their neighbourhood[1]. Instances eventually stored in memory are instances of which the nearest neighbours have *different* classifications. The assumption is that such instances mark the boundary of an area in which all instances are labeled with the same class; the instances that would be positioned in the centre of such areas in pure memory-based learning are not stored, since their position is safeguarded by the boundary instances surrounding it. This safety assumption makes IB2 a careful abstractor that may economise on memory usage considerably, but it also makes it sensitive to noise (Aha, Kibler, and Albert, 1991).

IB3 extends IB2, attempting to compensate for the fitting of noise. During incremental learning, of each stored instance records are updated on its successfulness as nearest neighbor

[1] Strictly speaking, editing in IB2 is not editing from memory, since left-out instances are only held temporarily in memory at the moment when their local statistics on whether they should be included are computed.

in the classification of all subsequently-stored instances (using some additional bootstrapping heuristics during the beginning stages of learning). On the basis of these records, a statistical test determines whether the instance should be regarded as noise and should be edited. An instance is edited when its classification accuracy is significantly lower than its class' observed frequency.

Although it may be profitable for generalisation accuracy to edit noise, it is crucial whether the estimation of the noisiness of instances does not mark productive instances (i.e., good classifiers for their own class) as noise, simply because they are in a small disjunct on their own. For some language learning tasks, among which grapheme-phoneme conversion and word pronunciation, the instance space is highly disjunct, and editing the smallest disjuncts almost immediately causes lower generalisation performance (Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming). For an instance in a small disjunct to be stored in memory in IB3 and not be deleted later on during learning, it is essential that it is at least once the nearest neighbour to an instance of the same class, which for very small disjuncts (e.g., those containing single instances) is unlikely to happen. IB3 may not be careful enough for these types of data.

2.1.2 Oblivious (partial) decision-tree abstraction

Many studies have demonstrated the positive effects of using informational abstraction, such as feature weighting, in the distance function (Cost and Salzberg, 1993; Wettschereck, Aha, and Mohri, 1997), on classification accuracy in memory-based learning of many types of tasks, including real-world learning tasks. This appears to hold for language learning tasks in general, as witnessed by several empirical studies (Weijters, 1991; Daelemans and Van den Bosch, 1992a; Van den Bosch, 1997). Daelemans and Van den Bosch (1992a) introduced feature weighting by computing their *information gain* (Quinlan, 1986). It appears profitable to include information-gain feature weighting in the distance function. Many other feature weighting methods exist (Salzberg, 1991; Kononenko, 1994; Wettschereck, Aha, and Mohri, 1997), but we focus here on information-gain (ratio) weighting as an example, and give a brief definition.

The function for computing information gain (henceforth IG) weighting looks at each feature of instances in an instance base in isolation, and measures how much information

it contributes to predicting the correct classification. The information gain of feature f is measured by computing the difference in entropy (i.e., uncertainty, or scrambledness of information) between the situations without and with knowledge of the value of that feature, as displayed in Equation 3:

$$w_f = \frac{H(C) - \sum_{v \in V_f} P(v) \times H(C|v)}{si(f)} \tag{3}$$

$$si(f) = -\sum_{v \in V_f} P(v) \log_2 P(v) \tag{4}$$

Where C is the set of class labels, V_f is the set of values for feature f, and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated from relative frequencies in the training set. The normalizing factor $si(f)$ (split info) is included to avoid a bias in favor of features with more values. It represents the amount of information needed to represent all values of the feature (Equation 4). The resulting IG values can then be used as weights $w_{i...n}$ in Equation 1.

Large differences in the IG of features are an adequate basis for inducing decision trees from instances, such as in C4.5 (Quinlan, 1986), or in IGTREE (Daelemans, Van den Bosch, and Weijters, 1997). We briefly discuss the latter algorithm, as it is included in the case studies described in Sections 3 and 4.

IGTREE is a method to optimise search in instance bases in memory-based learning (Daelemans, Van den Bosch, and Weijters, 1997), that compresses the content of the original instance base by careful abstraction. Abstraction takes the form of replacing feature-value information over sets of instances, along with classification information, by decision tree arcs and nodes. IGTREE builds *oblivious* decision trees, i.e., feature ordering is computed only at the root node and is kept constant during TDIDT, instead of being recomputed at every new node such as in C4.5 (Quinlan, 1993). Moreover, IGTREE does not prune low-frequency instances; it is only allowed to carefully abstract information redundant for the classification of the instances presented during training.

The basis for IGTREE construction is the ordering of features according to their IG. From a root node, labeled with the most frequent class in the instance base (i.e., the best guess on classification in the absence of information on feature values), arcs are generated labeled

with all occurring values of the feature with the *largest* IG. Each arc ends in a second-level node that represents the subset of instances with the specific value at the most important feature indicated by their arc, and that is labeled with the most frequently occurring class in that subset. This top-down induction of the decision tree is then applied recursively until a represented subset is labeled with a single class. Careful abstraction is applied to remove information that is redundant for classification: (i) no arcs and nodes are generated on less important features that were not inspected in a disambiguated path since they are redundant for classification, and (ii) leaf nodes that have the same class label as their parent node are also redundant, thus not stored. One may argue that each node construction constitutes yet another careful abstraction, since it summarises the presence of a feature value over many instances by only storing it once, along with a class or a class distribution. We assert that only the abstraction of a default class from an original distribution of classes in a non-ending node constitutes abstraction; feature-value summarisation on arcs is only an alternative (optimised) way of storing information that is not lost.

2.1.3 Carefully merged instances

Paths in decision trees can be seen as generalised instances, but in IGTREE and C4.5 this generalisation is performed up to the point where no actual instance is left in memory; all is converted to nodes and arcs. Counter to this strong generalisation by compression, approaches exist that start with storing individual instances in memory, and carefully merge some of these instances to become a single, more general instance, only when there is some evidence that this operation is not harmful to generalisation performance. Although overall memory is compressed, the memory still contains individual items on which the same k-NN-based classification can be performed. The abstraction occurring in this approach is that after a merge, the merged instances incorporated in the new generalised instance cannot be reconstructed individually. Example approaches to merging instances are NGE (Salzberg, 1991) and its batch variant BNGE (Wettschereck and Dietterich, 1995), and RISE (Domingos, 1996). We provide brief discussions of two of these algorithms: NGE and RISE.

NGE (Salzberg, 1991), an acronym for *Nested Generalised Exemplars*, is an incremental learning theory for merging instances (or exemplars, as Salzberg prefers to refer to instances

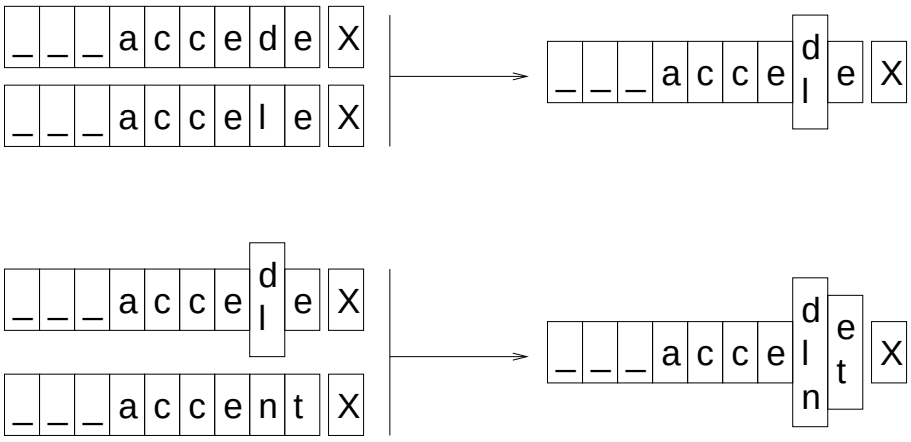

Figure 1: Two examples of the generation of a new hyperrectangle in NGE: from a new instance and an individual exemplar (top) and from a new instance and the hyperrectangle from the top example (bottom).

stored in memory) into *hyperrectangles*, a geometrically motivated term for merged exemplars. Partly analogous to IB2 and IB3, NGE[2] adds instances to memory in an incremental fashion (at the onset of learning, the memory is seeded with a small number of randomly-picked instances). Every time a new instance is presented, it is matched with all exemplars in memory, which can be individual or merged exemplars (hyperrectangles). When it is classified correctly by its nearest neighbour (an individual exemplar or the smallest matching hyperrectangle), the new example is merged with it, yielding a new, more general hyperrectangle.

Figure 1 illustrates two mergings of instances of the grapheme-phoneme conversion task with exemplars. On the top of Figure 1, the new instance __accele, labeled with class X (a code for the two-phoneme pronunciation /ks/), is merged with the single-instance exemplar __accede (also of class X), to form the generalised exemplar __acce{d,l}e mapping to their common class X (here, brackets denote the disjunction of values of a single feature). At the bottom of the figure, __acce{d,l}e–X is merged with __accent–X to form the more general __acce{d,l,n}{e,t}–X. *Abstraction* occurs because it is not possible to retrieve the individual instances that are nested in the generalised exemplar; new *generalisation* occurs because the generalised exemplar not only matches fully with its nested instances, but also matches fully with possible instances in which feature-value combinations occur that were not present in

[2]Salzberg (1991) makes an explicit distinction between NGE as a theory, and the learning algorithm EACH as the implementation; we will use NGE here to denote both.

the nested instances: it would also match __accene, __accelt, and __accedt perfectly.

Notice that with symbolic feature values, the geometrically intuitive concept of nesting becomes void. Since no real-valued distance is computed between symbolic feature values, but rather the simple all-or-none overlap similarity metric applies (Salzberg, 1991; Aha, Kibler, and Albert, 1991), merging yields flat *disjunctions of conjunctions* of feature values, as illustrated in Figure 1.

When a new instance is misclassified by its nearest neighbour, it is merged with the second-nearest neighbour if that neighbour would classify the new instance correctly (a "second-chance" heuristic, Salzberg (1991)). If not, the new instance is added to memory as an individual exemplar. It may be inside an existing hyperrectangle, thus representing an exception marked with a different class (a "hole") within the instance space bounded by that hyperrectangle.

Matching between new instances and (merged) exemplars in the implementation of NGE is augmented with two additional heuristics: (i) using the *class prediction strength* of an exemplar as a multiplication factor in the similarity function, and (ii) using incrementally-learned global feature weights set according to their contribution to classification error. For details on these weighting metrics the reader is referred to Salzberg (1991), and to Cost and Salzberg (1993) for an elaboration of the class-prediction strength metric.

RISE (*Rule Induction from a Set of Exemplars*) (Domingos, 1995; Domingos, 1996) is a unified multistrategy learning method, combining memory-based learning (viz. PEBLS, Cost and Salzberg (1993)) with rule-induction (Michalski, 1983; Clark and Niblett, 1989; Clark and Boswell, 1991). As in NGE, the basic method is that of a memory-based learner and classifier, only operating on a more general type of instance. RISE learns a memory filled with *rules* which are all derived from individual instances. Some rules are instance-specific, and other rules are generalised over sets of instances.

RISE inherits parts of the rule induction method of CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991). CN2 is an incremental rule-induction algorithm that attempts to find the "best" rule governing a certain amount of instances in the instance base that are not yet covered by a rule. "Goodness" of a rule is estimated by computing its apparent accuracy with

Laplacian correction (Niblett, 1987; Clark and Boswell, 1991). Rule induction ends when all instances are abstracted (covered by rules).

RISE induces rules in a careful manner, operating in cycles. At the onset of learning, all instances are converted to instance-specific rules. During a cycle, for each rule a search is made for the nearest instance not already covered by it that has the same class. If such an instance is found, rule and instance are merged into a more general rule. When identical rules are formed, they are joined. The assumption is made that performance retainment on the training set (i.e., the generalisation accuracy of the rule set on the original instances they were based on) also helps performance on test material to be retained at the level of the most accurate of its parent algorithms. At each cycle, the goodness of the rule set on the original training material (the individual instances) is monitored. RISE halts when this accuracy measure does not improve (which may already be the case in the first cycle, yielding a plain memory-based learning algorithm). In a series of experiments it is shown that RISE can improve on its memory-based parent PEBLS, as well as on its rule-induction parent CN2, on a significant number of benchmark learning tasks (Domingos, 1996).

Applied to symbolically-valued data, RISE creates rules that are left-hand side conjunctions of *conditions* coupled with a right-hand side consequent being the rule's class. A condition couples a feature to one value in an equality. A rule may contain only one condition per feature, and may contain no conditions at all. Figure 2 illustrates the merging of individual instances into a rule. The rule contains seven non-empty conditions, and two empty ones (filled with wildcards, '*', in the figure). The rule now matches on every instance beginning with _acce, and receives a goodness score (i.e., its apparent accuracy on the training set with Laplacian correction) of 0.095.

Instance classification is done by searching for the best-matching rule, always selecting the rule with the highest Laplacian accuracy (Clark and Boswell, 1991). As a heuristic add-on for dealing with symbolic values, RISE incorporates a value-difference metric (Stanfill and Waltz, 1986; Cost and Salzberg, 1993) by default, called the *simplified value-difference metric* (SVDM) due to its simplified treatment of feature-value occurrences in the VDM function (Domingos, 1996).

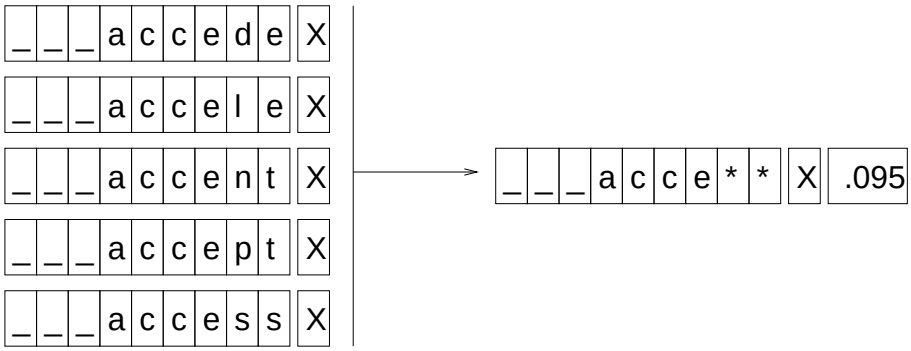

Figure 2: An example of an induced rule in RISE, displayed on the right, with the set of instances that it covers (and from which it was generated) on the left.

2.2 FAMBL: merging instance families

FAMBL, for *FAMily-Based Learning*, is a new algorithm that constitutes an alternative approach to careful abstraction over instances. The core idea of FAMBL is to transform an instance base into a set of *instance family expressions*. An instance family expression is a hyperrectangle, but the procedure for merging instances differs from that in NGE or in RISE. First, we outline the ideas and assumptions underlying FAMBL. We then give a procedural description of the learning algorithm.

2.2.1 Instance families

Classification of an instance in memory-based learning first involves a search for the nearest neighbours of that instance. The value of k in k-NN determines how many of these neighbours are used for extrapolating their (majority) classification to the new instance. A fixed k ignores (smoothes) the fact that an instance is often surrounded in instance space by a number of instances of the same class that is actually larger or smaller than k. We refer to such variable-sized set of same-class nearest neighbours as an instance's *family*. The extreme cases are (i) instances that have a nearest neighbour of a different class, i.e., they have no family members and are a family on their own, and (ii) instances that are have as nearest neighbours all other instances of the same class.

Thus, families are class clusters, and the number and sizes of families in a data set reflect the *disjunctivity* of the data set: the degree of scatteredness of classes into clusters. In real-world data sets, the situation is generally somewhere between the extremes of total

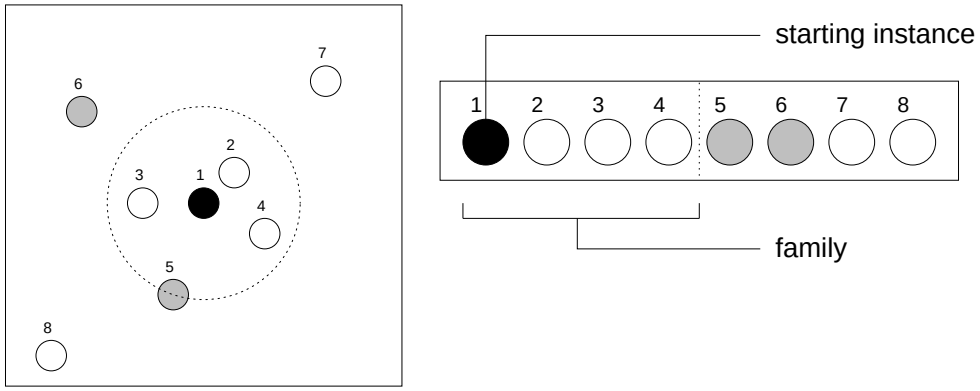

Figure 3: An example of a family in a two-dimensional instance space (left). The family, at the inside of the dotted circle, spans the focus instance (black) and the three nearest neighbours labeled with the same class (white). When ranked in the order of distance (right), the family boundary is put immediately before the first instance of a different class (grey).

disjunctivity (one instance per cluster) and no disjunctivity (one cluster per class). Many types of language data appear to be quite disjunct (Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming). In highly disjunct data, classes are scattered among many small clusters, which means that instances have few nearest neighbours of the same class on average.

Figure 3 illustrates how a family of an instance is determined in a simple two-dimensional example instance space. All nearest neighbours of a randomly-picked starting instance (marked by the black dot) are searched and ranked in the order of their distance to the starting instance. Although there are five instances of the same class in the example space, the family of the starting instance contains only three instances, since its fourth-nearest instance is of a different class.

Families are converted in FAMBL to *family expressions*, which are hyperrectangles, by merging all instances belonging to that family simultaneously. Figure 4 illustrates the creation of a family expression from an instance family. In contrast with NGE,

- family expressions are created in one operation, rather than by step-wise nesting of each individual family member.
- a family is abstracted only once and is not merged later on with other instances or family expressions.
- families cannot contain "holes", i.e., instances with different classes, since the definition of family is such that family abstraction halts as soon as the nearest neighbour with a

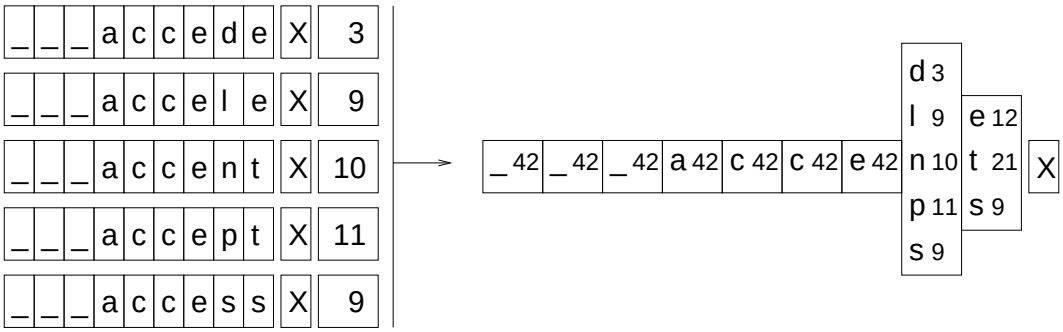

Figure 4: An example of family creation in FAMBL. Four grapheme-phoneme instances, along with their token occurrence counts (left) are merged into a family expression (right). Feature-value occurrences are merged and stored along.

different class is met in the local neighbourhood.

- FAMBL keeps track of all occurrence counts of feature values in families, assuming that this occurrence information may help in resolving ties in nearest-neighbour search. It can be seen in Figure 4 that occurrences of feature values of instance tokens are merged along with the values themselves.
- FAMBL is non-incremental: it operates on a complete instance base stored in memory.

Notice that families reflect the locally optimal k surrounding the instance around which the family is created. The locally optimal k is a notion that is also used in locally-weighted learning methods (Vapnik and Bottou, 1993; Atkeson, Moore, and Schaal, 1997); however, these methods do not abstract from learning material. In this sense, FAMBL can be seen as a locally-weighted abstractor.

2.2.2 The FAMBL algorithm

The FAMBL algorithm has a learning component and a classification component. The learning component of FAMBL is composed of two stages: a *probing* stage and a *family extraction* stage.

The probing stage is a preprocessing stage to the actual family extraction as outlined above. The reason for preprocessing is visualised in Figure 5. The random selection of instances to be a starting point for family creation can be quite unfortunate. When, for example, the middle instance in the left part of Figure 5 is selected first, a seven-instance family is formed with relatively large within-family distances. Moreover, three other instances that are actually

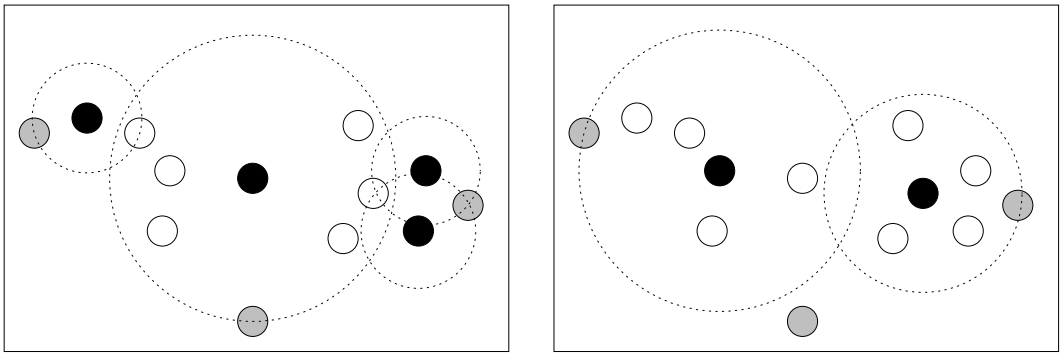

Figure 5: Illustration of the need for the preprocessing stage in FAMBL. The left figure shows a big family with seven members, forcing the remaining three instances to be their own family. The right figure shows the same space, but with two other starting points for family creation, displaying a more evenly divided space over two families. Black instances denote starting points for family creation; white instances are of the same class as the starting points, and grey instances are of a different class.

quite close to members of this big family become isolated and are necessarily extracted later on as single-instance families. The situation in the right part of Figure 5 displays a much more desirable situation, in which the space is more evenly divided between only two families instead of four.

In the probing stage, all families are extracted randomly and straightforwardly, while records are maintained of (i) the size of each family, and (ii) the average distance between the starting instance of each family and the other instances in the family. When all instances are captured in families, the mean and the median of both records are computed, and *both medians are used as threshold values for the second, actual family extraction phase.* This means that in the family extraction phase,

1. no family is extracted that has more members than the probed median number of members.
2. no family is extracted that has an average distance from the starting instance to the other family members larger than the probed median value.

Thus, the actual family extraction phase applies extra careful abstraction, under the assumption that it is better to have several medium-sized, adjacent families of the same class than one big family overlapping the medium ones except for some adjacent boundary instances that get isolated.

Procedure FAMBL PROBING PHASE:

Input: A training set TS of instances $I_{1...n}$, each instance being labeled with a family-membership flag

Output: Median values of family size, M_1, and within-family distance, M_2

$i = 0$

1. Randomize the ordering of instances in TS
2. Set the family-membership flags of all instances to $FALSE$
3. While not all family flags are $TRUE$, Do
   - While the family-membership flag of I_i is $TRUE$ Do increase i
   - Compute NS, a ranked set of friendly nearest neighbours to I_i among all instances with family-membership flag $FALSE$. Nearest-neighbour instances of a different class with family-membership flag $TRUE$ are still used for determining the boundaries of the family.
   - Record the number of members in the new virtual family: $|NS| + 1$
   - Record the average distance of instances in NS to I_i
   - Set the membership flags of I_i and all instances in NS to $TRUE$
4. Compute M_1
5. Compute M_2

Figure 6: Schematised overview of the probing phase in FAMBL.

It should be noted that with symbolic feature values (and without value-difference metrics), the k nearest neighbours in any matching operation may not be the same as the number of different distances found in this nearest-neighbour set. For example, five nearest neighbours may be equally-best matching with an new instance when they all differ in the same single feature value. The k in FAMBL, including the median value found in the probing phase, is chosen to refer to the k number of different *distances* found among the nearest neigbours.

A schematised overview of the two FAMBL learning phases is displayed in Figures 6 and 7.

After learning, the original instance base is discarded, and further classification is based only on the set of family expressions yielded by the family-extraction phase. Classification in FAMBL works analogously to classification in pure memory-based learning, and classification in NGE: a match is made between a new test instance and all stored family expressions. When a family expression records a disjunction of values for a certain feature, matching is perfect when one of the disjuncted values matches the value at that feature in the new instance. Merged feature-value counts are summed for each feature-value match, to be used in case of ties: when two or more family expressions of different classes match equally well with the new instance, the expression is selected with the highest summed occurrence of matched features.

Procedure FAMBL FAMILY-EXTRACTION PHASE:

Input: A training set TS of instances $I_{1...n}$, each instance being labeled with a family-membership flag

Output: A family set FS of family expressions $F_{1...m}$, $m \leq n$

$i = f = 0$

1. Randomize the ordering of instances in TS
2. Set the family-membership flags of all instances to $FALSE$
3. While not all family flags are $TRUE$, Do
   - While the family-membership flag of I_i is $TRUE$ Do increase i
   - Compute NS, a ranked set of friendly nearest neighbours to I_i among all instances with family-membership flag $FALSE$. Nearest-neighbour instances of a different class with family-membership flag $TRUE$ are still used for determining the boundaries of the family.
   - Compute the number of members in the new virtual family: $|NS| + 1$
   - Compute the average distance of all instances in NS to I_i: $A_{NS,I}$
   - While $((|NS + 1| > M_1) OR (A_{NS,I} > M_2))$ Do remove the most distant family member to I in NS
   - Set the membership flags of I_i and all remaining instances in NS to $TRUE$
   - Merge I_i and all instances in NS into the family expression F_f
   - $f = f + 1$

Figure 7: Schematised overview of the family-extraction phase in FAMBL.

When the tie remains, the class is selected that occurs the most frequently in the complete family expression set.

We conclude our description of the FAMBL algorithm by noting that FAMBL allows for the inclusion of informational abstraction in the form of feature-weighting, instance-weighting and value-difference metrics. For comparisons with other algorithms, as described in the next section, we have included some of these metrics as options in FAMBL. Weighting metrics are likely to have a profound effect on family extraction. For example, a study by Van den Bosch (1997) suggests that using information-gain feature weighting (Quinlan, 1986) in pure memory-based learning (viz. IB1-IG, Daelemans and Van den Bosch (1992a)), can yield considerably bigger families.

3 Effects of careful abstraction: A comparative case study

We performed a series of experiments concerning the application of a wide range of careful-abstracting methods, described in the previous section, to grapheme-phoneme conversion. It is intended to provide indications for the efficacy of different careful abstraction approaches as compared to pure memory-based learning, including variants of both approaches which use weighting metrics. As said, the chosen benchmark task is known for its sensitivity to abstraction, so that it is likely that any differences in abstraction methods show up most clearly in results obtained with this task.

From an original instance base of 77,565 word-pronunciation pairs extracted from the CELEX lexical data base of Dutch (Baayen, Piepenbrock, and van Rijn, 1993) we created ten equal-sized data sets each containing 7,757 word-pronunciation pairs. Using windowing (cf. Section 2) and partitioning of this data in 90% training and 10% test instances, ten training and test sets are derived containing on average 60,813 and 6761 instances, respectively. These are token counts; in the training sets, 54,295 instance types occur (on average). Notice that these training and test sets are not constructed as is usual in 10-fold cross validation (Weiss and Kulikowski, 1991), where training sets largely overlap. Here, each training and test set is derived from words that are not used in any other training or test set[3]. This approach, using relatively small data sets as compared to earlier studies (Van den Bosch, 1997), was taken to cope with the extreme memory demands of some of the algorithms tested. Furthermore, we believe that the approach alleviates some of the objections raised against using t-tests for significance testing with dependent data sets (Salzberg, 1997; Dietterich, 1998).

In this series, one experiment consists of applying one algorithm to each of the ten training sets, and a test on each of the respective test sets[4]. Apart from the careful abstractors IB2, IB3, IGTREE, RISE, NGE, and FAMBL, we include the pure memory-based learning algorithms PEBLS (Cost and Salzberg, 1993) and IB1(-IG) (Aha, Kibler, and Albert, 1991; Daelemans

[3] Nevertheless, since words are partly similar and since windowing only looks at parts of words, there is some overlap in the instances occurring in the different sets.

[4] Experiments with IB1, IB1-IG, and IGTREE were performed using the TiMBL software package, available at `http://ilk.kub.nl/`.

algorithm	metrics IG	CPS	VDM	CN2	generalisation error (%)	±	number of memory items
PEBLS			x		88.99	0.32	54295
IB1	x		x		88.91	0.63	54294
RISE			x	x	88.88	0.61	20252
IB1	x				88.83	0.61	54294
FAMBL	x				88.82	0.61	35141
IGTREE	x				88.46	0.53	12202
IB1			x		87.88	0.66	54294
PEBLS		x	x		87.51	0.60	54294
IB1					78.07	0.79	54294
IB2					74.27	0.66	19917
IB3					73.33	0.45	19196
FAMBL					72.26	0.76	31889
NGE		x			61.79	0.86	25627
C4.5	x				87.10	0.43	20889
Naive Bayes					74.16	0.58	–

Table 2: Overview of generalisation errors and memory usage obtained with pure memory-based learning and careful abstraction methods, as well as with C4.5 and Naive Bayes.

and Van den Bosch, 1992b) in the comparison. Each experiment yields (i) the mean generalisation accuracy, in percentages correctly classified test instances, (ii) a standard deviation on this mean, and (iii) a count on the number of items in memory (i.e., instances or merged instances). Table 2 lists these experimental outcomes for all algorithms tested. As some of these algorithms use metrics for weighting (IG, class-prediction strength (CPS), value-difference metrics (VDM), or rule-induction metrics in RISE (the CN2 metrics), we have marked the use of such metrics explicitly in separate columns, using 'x' for denoting the presence of a metric in the algorithm in that row. Table 3 displays the significance test results (one-tailed t-tests) performed with each pair of algorithms. Significant results (i.e., with $p < 0.05$, marked with asterisks) indicate that the algorithm in the row has a significantly higher generalisation accuracy than the algorithm in the column.

The results indicate a group of five best-performing algorithms that, in pair-wise comparisons, do not perform significantly different: PEBLS (with MVDM), (ii) IB1 (with IG and MVDM), (iii) RISE (with SVDM and the CN2 rule-induction metric), (iv) IB1 (with IG), and (v) FAMBL (with IG). Moreover, IGTREE does not perform significantly different with all algorithms in this group except for PEBLS. A general property of this group is that they all employ weighting metrics. All algorithms using IG weighting are in this best-performing group, when IGTREE is included. Thus, these results add to the existing empirical evidence of positive effects of

algorithm	IB1 IG MVDM	RISE SVDM CN2	IB1 IG	FAMBL IG	IGTREE	IB1 MVDM	PEBLS CPS MVDM	IB1	IB2	IB3	FAMBL	NGE CPS
PEBLS-CPS-MVDM					**	***	***	***	***	***	***	***
IB1-IG-MVDM	—					**	***	***	***	***	***	***
RISE-MVDM-RI	—	—				**	***	***	***	***	***	***
IB1-IG	—	—	—			**	***	***	***	***	***	***
FAMBL-IG	—	—	—	—		**	***	***	***	***	***	***
IGTREE	—	—	—	—	—	*	***	***	***	***	***	***
IB1-MVDM	—	—	—	—	—	—		***	***	***	***	***
PEBLS-CPS-MVDM	—	—	—	—	—	—	—	***	***	***	***	***
IB1	—	—	—	—	—	—	—	—	***	***	***	***
IB2	—	—	—	—	—	—	—	—	—	***	***	***
IB3	—	—	—	—	—	—	—	—	—	—	***	***
FAMBL	—	—	—	—	—	—	—	—	—	—	—	***

Table 3: Significance results: one-tailed t-test outcomes for algorithm pairs, expressing whether the algorithm in the row has a significant lower generalisation error than the algorithm in the column. '*' denotes $p < 0.05$; '**' denotes $p < 0.01$; '***' denotes $p < 0.001$.

weighting metrics.

Of course, our focus is on the effects of careful abstraction. Within the group of five best-performing algorithms, two are careful abstractors, performing careful merging of instances: RISE and FAMBL. RISE is able to shrink its rule base down to 20,252 rules. As compared to the 54,295 instance types that are all stored in pure memory-based learning (PEBLS, IB1), RISE obtains an item compression of 62.7%. FAMBL compresses less: 35.3%. IGTREE, boundary member of the best-performing group, is able to compress the instance bases into trees that contain, on average, only 12,202 nodes, which are counted as items here, but which do not directly relate to instances. In sum, we see the careful abstraction approaches RISE, FAMBL, and IGTREE reducing the number of memory items, while equalling the performance of pure memory-based approaches, given a specific choice of weighting metrics.

It is relevant to our central topic to analyse the effects of careful abstraction *without* additional weighting, thus drawing on the results obtained with IB1, IB2, IB3, and FAMBL without weighting. Here, pure memory-based learning is at a significant advantage: IB1 is significantly more accurate than each of the three carefully-abstracting methods. In turn, IB2 performs significantly better than IB3 and FAMBL, and IB3 performs significantly better than FAMBL. For this task, (i) editing in IB2 and IB3 is harmful; (ii) the noise reduction in IB3 is extra harmful; and (iii) family extraction is even more harmful than incremental editing. For FAMBL, this provides an indication that IG weighting may be essential for the success of the

approach. In other words, IG weighting yields a rescaling of the instance space better suited for generalisation.

Finally, we make two side observations. First, NGE with its default class-prediction strength weighting, performs significantly worse than IB2, IB3, and FAMBL. It is unclear what contributes most to this low accuracy: its strategy to nest hyperrectangles, or the class-prediction strength weighting. Since PEBLS with class-prediction strength weighting attains a significantly lower accuracy than PEBLS without this weighting, NGE's performance may be also be suffering from this weighting. Second, when we compare the performances of the memory-based learning algorithms with decision-tree learning in C4.5 and the Naive Bayes classifier (Langley, Iba, and Thompson, 1992) (displayed at the bottom of Table 2), we can determine roughly that C4.5 performs worse than most weighted memory-based methods, a result that is in agreement with Van den Bosch (1997) and Daelemans, Van den Bosch, and Zavrel (1998 forthcoming), and that the Naive Bayes classifier, which may serve as a good baseline classifier, performs at the level of IB2.

4 A comparative study on a range of language learning tasks

The dataset of grapheme-phoneme conversion task instances used in Section 3 is kept artificially small. For this task, but also for other language tasks, considerably larger data sets are available. It is not uncommon in inductive language learning studies to use data sets with hundreds of thousands of instances, first, because learning curve studies show generalisation improvements at very large data set sizes (Daelemans, Berck, and Gillis, 1997; Van den Bosch, 1997); second, because many language data sets represent sparse instance spaces: many samples are needed to represent at least a fair share of normally distributed data for the task at hand.

Many current implementations of careful-abstraction learning algorithms do not allow for experiments on data sets of this size, although (i) optimisations may resolve part of the problem, and (ii) computer technology developments continue to soften the need for optimisations.

We were able to apply FAMBL to large data sets however, and present the results of these experiments here. We performed a series of 10-fold cross validation experiments (Weiss and Kulikowski, 1991) with FAMBL, augmented with IG feature weighting, on grounds of the positive effect of this metric on FAMBL's performance on the grapheme-phoneme conversion task reported in Section 3. FAMBL is applied to language data sets used in earlier studies (Van den Bosch, Daelemans, and Weijters, 1996; Van den Bosch, 1997; Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming). The data sets represent a range of language tasks: morphological segmentation, word pronunciation, base-NP chunking, and PP attachment. Table 4 lists the numbers of instances, feature values, and classes of the data sets of these four tasks. We briefly outline the underlying tasks that these data sets represent.

Morphological segmentation (henceforth MS) is the segmentation of words into labeled morphemes. Each instance represents a window snapshot of a word of nine letters. Its class represents the presence or absence of a morpheme boundary immediately before the middle letter. If present, it also encodes the type of morpheme starting at that position, i.e., whether it is a stem, an inflection, a stress-neutral affix, or a stress-affecting affix. For example, the word booking is composed of the stem book and the inflection ing; consequently, the first instance generated from the word is ___booki with class 'present–stem', the second __bookin with class 'absent', the fifth booking__ with class 'present-inflection', the sixth ooking__ with class 'absent', etc. See (Van den Bosch, Daelemans, and Weijters, 1996) for more details.

Word pronunciation is similar to the grapheme-phoneme conversion task illustrated earlier, but with two differences: (i) the windows only span seven letters, and (ii) the class represents a combined phoneme and a stress marker. The stress marker part denotes whether the phoneme is the first of a syllable receiving primary or secondary stress. For example, class '/b/1' indicates a phoneme /b/ that is the first phoneme of a syllable receiving primary stress, which would be the class label of the instance __book from the word booking. See (Van den Bosch, 1997) for more details. The task is referred to as GS for Grapheme-phoneme conversion plus Stress assignment.

Base-NP chunking (henceforth NP) is the segmentation of sentences into non-recursive

NPs (Abney, 1991). Veenstra (1998 forthcoming) used the Base-NP tag set as presented in (Ramshaw and Marcus, 1995): *I* for inside a Base-NP, *O* for outside a Base-NP, and *B* for the first word in a Base-NP following another Base-NP. As an example, the IOB tagged sentence: "The/I postman/I gave/O the/I man/I a/B letter/I ./O" results in the following Base-NP bracketed sentence: "[The postman] gave [the man] [a letter]." The data are based on the same material as used by Ramshaw and Marcus (1995) which is extracted from the Wall Street Journal text in the parsed Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993). An instance (constructed for each focus word) consists of features referring to words (two left-neighbour and one right-neighbour word), their part-of-speech tags, and IOB tags (predicted by a first-stage classifier) of the focus and the two left and right neighbour words. See Veenstra (1998 forthcoming) for more details, and (Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming) for a series of experiments on the data set also used here.

PP attachment (henceforth PP) is the attachment of a PP in the sequence VP NP PP (VP = verb phrase, NP = noun phrase, PP = prepositional phrase). The data consists of four-tuples of words, extracted from the Wall Street Journal Treebank (Ratnaparkhi, Reynar, and Roukos, 1994). They took all sentences that contained the pattern VP NP PP and extracted the head words from the constituents, yielding a V N1 P N2 pattern (V = verb, N = noun, P = preposition). For each pattern they recorded whether the PP was attached to the verb or to the noun in the treebank parse. For example, the sentence *"he eats pizza with a fork"* would yield the pattern eats, pizza, with, fork, verb.. A contrasting sentence would be *"he eats pizza with anchovies"*: eats, pizza, with, anchovies, noun. From the original data set, used in statistical disambiguation methods by Ratnaparkhi, Reynar, and Roukos (1994) and Collins and Brooks (1995), and in a memory-based learning experiment by Zavrel, Daelemans, and Veenstra (1997), (Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming) took the train and test set together to form the data also used here.

For each task FAMBL is compared with IB1, IB1-IG, and IGTREE. Table 5 lists the generalisation accuracies obtained in these comparisons, on the four tasks. The results of IB1-IG

Task	# Features	# Values of feature 1	2	3	4	5	6	7	8	9	10	11	# Classes	# Data set instances
MS	9	42	42	42	42	41	42	42	42	42			2	573,544
GS	7	42	42	42	41	42	42	42					159	675,745
PP	4	3,474	4,612	68	5,780								2	23,898
NP	11	20,231	20,282	20,245	20,263	86	87	86	89	3	3	3	3	251,124

Table 4: Specifications of the four investigated data sets of the M, GS, PP, and NP learning tasks: numbers of features, values per feature, classes, and instances.

and IGTREE are reproduced from (Daelemans, Van den Bosch, and Zavrel, 1998 forthcoming). One-tailed t-tests yield significance results that show at a general level that (i) IB1-IG is significantly more accurate than IGTREE on all tasks; (ii) FAMBL is significantly more accurate than IGTREE on the MS, NP, and PP tasks, and (iii) IB1-IG is significantly more accurate than FAMBL on the MS, GS, and PP tasks. The latter results show that FAMBL's careful abstraction is not careful enough on these tasks.

Task	IB1-IG %	±	>FAMBL?	>IGTREE?	FAMBL-IG %	±	>IGTREE?	IGTREE %	±
MS	98.02	0.05	***	***	97.84	0.06	***	97.45	0.06
GS	93.45	0.15	**	***	93.22	0.24		93.09	0.15
NP	98.07	0.05		***	98.04	0.05	***	97.28	0.08
PP	83.48	1.16	**	***	81.80	1.14	***	78.28	1.79

Table 5: Generalisation accuracies (percentages correctly classified test instances, with standard deviations) of IB1-IG, FAMBL, and IGTREE on the MS, GS, NP, and PP tasks. Asterisks denote the outcomes of one-tailed t-tests, denoting a significantly better accuracy of the algorithm in the row compared to the algorithm in the column. '**' denotes $p < 0.01$; '***' denotes $p < 0.001$.

On closer inspection of the experimental results, the behaviour of FAMBL on the four tasks turns out to be different. We monitored for all experiments the number of families that FAMBL probed and extracted, including the median sizes and within-family distances it found during probing. Table 6 displays these results averaged over the ten experiments performed on each task. The table also displays two additional quantities: (i) the measured *clusteredness*, which reflects the number of disjunct clusters (families) per class, averaged over classes, weighted by their frequency, and (ii) the percentage of compression over the number of memory items (instances vs. family expressions). Both quantities are reported for the probing stage as well as the family stage.

Table 6 illustrates how much abstraction is actually attained by FAMBL. In the probing

Task	generalisation accuracy %	±	probed families #	cluster-edness	% item compr.	k	median distance	extracted families #	cluster-edness	% item comp.
MS	97.84	0.06	18,783	8,435	93.4	2	0.078	131,776	74,616	53.7
GS	93.22	0.24	37,457	1,693	83.2	1	0.084	153,441	8,445	31.0
NP	98.04	0.05	5,238	2,413	97.7	3	0.155	59,376	29,072	72.4
PP	81.80	1.14	157	78	99.3	1	0.078	6,414	3,193	70.1

Table 6: Specifications of the average number of families extracted by FAMBL on each of the four learning tasks, with mean and median family sizes (k) and within-family distance. The generalisation performance is repeated from Table 5.

phases, the clusteredness of classes is already in the order of a thousand, except for the PP task, while in the family extraction phase clusteredness in the MS and NP tasks reaches levels in the order of ten thousand. The numbers of extracted families are also very high (e.g., 153,441 for the GS task in the family extraction phase). The increased numbers of families and the clusteredness in the family extraction phase as compared to the probing phase are the direct effect of using the thresholds computed in the probing phase. The thresholds on the k, for example, illustrate that family extraction is strictly limited to $k = 1$ in the GS and PP tasks, i.e., family members are allowed to mismatch in only one feature value. With MS (2) and NP (3), mismatching of family members is slighly more liberal.

In the extraction phase, compression (the percentage reduction on the number of items in memory, from instances in pure memory-based learning to family expressions) ranges from 31.0% with the GS task to 72.4% with NP, which is considerable. The lowest compression is obtained with GS, on which FAMBL did not outperform IGTREE, and the highest compression is obtained with NP, on which FAMBL equalled with IB1-IG. This would suggest that the underlying assumptions of FAMBL apply successfully to the NP task, and that the GS task data has properties that FAMBL is less prepared to handle adequately. We have two working hypotheses on what these properties might be:

1. The GS data is very disjunct: FAMBL detects a relatively high number of families during probing and family extraction. The random selection of starting points for family extraction, although heuristically patched with the preprocessing of the probing phase, may still lead to unwanted effects as illustrated in Figure 5 when data is very disjunct.

2. FAMBL tends to blur *feature interaction*: it allows the combination of feature values that

never occurred in that constellation in the learning material, while for some tasks, including GS, this generalisation may be unwanted. For example, considering the example of Figure 3, it may be actually counterproductive for the family expression in this figure to fully match with __accepe or __accedt, which are nonsensical, but for which it is in any case unclear whether the cc would be pronounced /ks/.

Feature interaction on a local basis is ignored in all memory-based learning methods mentioned in this paper; we return to this matter in Section 5 in which we discuss openings to add feature-interaction methods in memory-based learning.

While our focus is on language learning, FAMBL is a machine learning algorithm that may also be tested according to more standard machine learning methodology. In machine learning it is common to compare algorithms on series of benchmark data sets of very different nature, to avoid the comparison of algorithms on data on which one is tuned to, and the other is not typically suited for (Salzberg, 1997). FAMBL may have a bias to language-like data. Only a small part of typical benchmark data sets is language-related. We have applied FAMBL to a selection of benchmark tasks from the UCI repository (C. Blake and Merz, 1998) with only symbolic feature values, using tenfold cross-validation experiments. Table 7 shows the results from the probing phase, repeating some of the results obtained with the language data sets earlier. In terms of clusteredness and numbers of probed families, only the PP data is near some of the benchmark data sets. The other three language learning task data sets are so much bigger that any further comparisons with benchmark data sets become blurred by this factor. Although we plan to pursue investigating any task bias of FAMBL, benchmark data will need to be found that at least approach the data set sizes of our language data. The generation of artificial data may be needed (Aha, 1992).

5 Discussion

We have reported on two case studies of applying abstraction methods in memory-based learning in a careful manner, to language learning tasks. In a first study, on learning grapheme-phoneme conversion from a moderately-sized data set, we found that the careful abstractors RISE, FAMBL, and IGTREE were able to equal the generalisation accuracy of their pure memory-

Task	data set size	# probed families	cluster-edness	median k	% memory item compr.
MS	573,544	18,783	8435	2	93.4
GS	675,745	37,457	1693	1	83.2
NP	251,124	5,238	2413	3	97.7
PP	23,898	157	78	1	99.3
audiology	226	72	7	1	60.5
kr vs kp	3,198	137	69	3	95.2
mushroom	8,124	23	11	40	99.7
nursery	12,961	682	198	2	94.2
soybean-l	683	90	9	2	84.3
splice	3,190	334	128	3	87.7
tic-tac-toe	958	161	84	3	81.4
votes	435	38	19	1	87.7

Table 7: Comparison of data set size, average numbers of probed families, clusteredness, median family size (k), and memory item compressions, between the language data and a selection of symbolic UCI benchmark data, as measured in FAMBL's probing phaseq (averaged over 10-fold cross validation experiments).

based counterparts PEBLS and IB1-IG. All best-performing algorithms implement (combinations of) weighting metrics; without them, any careful abstraction is harmful to generalisation performance as compared to pure memory-based learning.

In a second case study we applied the pure memory-based learning algorithm IB1-IG and the careful abstractors FAMBL and IGTREE to a range of language learning tasks (reproducing some of the work presented in Daelemans, Van den Bosch, and Zavrel (1998 forthcoming)). While IGTREE, which creates oblivious decision trees, performed worst overall, thus displaying harmful abstraction, FAMBL performed closer to IB1-IG, though only equalling it on only one task (base-NP chunking). Closer analyses of the learned models indicated that tasks such as grapheme-phoneme conversion and word pronunciation may have properties (such as local feature interaction) that FAMBL does not handle adequately.

Although the obtained results do not point at the superfluousness of pure memory-based learning, we have seen that carefully transforming an instance base into a set of generalised instances may yield compact models that perform close or equal to their pure memory-based counterparts. Careful abstraction in current approaches turns out to be not careful enough sometimes, and these current approaches may be failing to detect fine-grained and local feature interactions. Nevertheless, we believe these problems may be tackled within the framework of careful abstraction, yielding a general approach that may demonstrate that generalised

instances can be working units in memory-based learning and processing.

In addition, we note, qualitatively and briefly, some interesting features of family-based learning that allow for further research. Consider the examples displayed in Table 8 of actual family expressions as found by FAMBL on the five language learning tasks investigated here. For each task, three examples are given. Curly brackets bound the disjunctions of merged values at single features. We note two general characteristics of families we see being extracted by FAMBL:

1. In every family, there is at least one feature that has one fixed value. In some examples, most or all features are fixed (e.g, in the 'dioxide' example of the M3 task, apparently disambiguating the segmentation between i and o in this particular word with other ...io... sequences in which this segmentation does not occur). In general, families are always characterised by at least one fixed feature value, which is usually a feature with a high (or the highest) information gain.

2. Values that are grouped on one feature are often related (just as they will have, on average, small value difference, as value difference metrics compute precisely such class-related cooccurrences as happening in families, only on a global level). In cases where values are letters, graphematically and phonetically close groups, such as {a,e,i} or {a,o,u}, tend to reoccur. In cases where values are words, grouped values often appear to display some sort of syntactic–semantic relatedness.

In sum, there appears to be information hidden in extracted families that may be useful for other purposes, or for further abstraction. We now turn to our final remarks on future topics of research that elaborate on these indications.

5.1 Future research

Summarising, we identify four strands of future research that we view as relevant follow-ups of the case studies described in this article and other work on abstraction in memory-based learning. First, from the language learning perspective:

- Families extracted in FAMBL show interesting grouping of values that may be generalised further, e.g., by summarising frequently reoccurring groups using single identifiers or

Task	Example family expression	class
GP	_ {b,p,c,t,w} r i c k {k,l,i,s} {i,n,_} {l,e,g,_}	-
	{o,e,i} {u,s} {s,l,t} n e s s _ _	I
	_ _ _ r e {w,o} {r,i} {i,r} {t,e}	i
MS	_ _ _ _ u n {d,s} i {g,d,s,v}	2
	_ _ d i o x i d e	c
	{j,u,r,h} {i,a} {g,n} g l i e r _	0
GS	{_,n,a} {n,c} o n - {f,v} {i,l,o}	0n
	{e,f,g,i,k,l,m,p,r,s,u,v} e t e d _ _	0I
	{_,o,y} s {y,a} n t {h,a} {e,c,x}	0n
NP	the {news,notion,time,understanding} that {British, Mr.} DT NN IN NP I I I	O
	{of,solar} {vans,systems,reporters} and {light,TV} {IN,JJ} NNS CC NN I I {O,I}	O
	{sluggish,possible,second} {growth,sale,quarter} or {even,other,second} JJ NN CC JJ I I I	I
PP	{taken,'s,casts,has,is,play} {case,nothing,light,sketches,number,outfielder} on side	V
	boost stake in {conglomerate,business,maker}	N
	adding {confusion,argument,insult,land,measures,money,penny,voices} to ...	
	... {situation,arsenal,injury,residence,it,balances,tax,chorus}	V

Table 8: Examples of probed families for each of the five language learning tasks investigated in Sections 3 and 4. GP refers to grapheme-phoneme conversion (Section 3). Occurrence counts of feature values are left out for clarity.

wildcards. Moreover, merged value groups represent a sort of non-hierarchical clustering, that may be used as (or transformed into) an information source for the learning task itself, or to other related learning tasks.

- Memory-based learning in the approaches mentioned in this article all ignore feature interaction: the phenomenon of feature combinations that also contribute significant information to classification when taken together. This may also refer to exploring the interaction of specific values of different features. The phenomenon is addressed in recent work on maximum entropy models applied to language learning (Ratnaparkhi, 1996; Ratnaparkhi, 1997) and Winnow algorithms (Golding and Roth, 1998 forthcoming). We view the incorporation of feature interaction in memory-based learning as feasible, certainly when an integrative approach is taken combining memory-based learning with these related areas.

As regards the learning and classification algorithms presently constituting FAMBL, we identify the following two topics of further research as most relevant:

- Asymptotic analyses of storage, learning, and classification in FAMBL need to be made, and at least compared with those for the related approaches RISE and NGE. FAMBL's current implementation is fast, but empirical comparisons of learning speed of differ-

ent algorithms with different levels of optimisations in their implementations does not constitute sound data for concluding that FAMBL is also theoretically more efficient.

- Investigations should be performed into equipping FAMBL with less random and more sensible-heuristic-driven (yet non-parametric) probing and selection of families, provided that the current speed of FAMBL is not harmed seriously. It is important to investigate what unwanted effects may be caused by FAMBL's random-selection approach to probing and extraction of families, and how to possibly counteract them.

While a first target of these future investigations will be to make FAMBL handle certain types of data more adequately, we hope to arrive at an integrative view on careful abstraction and the nature of the basic storage units in memory-based learning in general, also connecting to maximum-entropy, Winnow, and possibly also connectionist approaches to abstraction from full memory storage.

Acknowledgements

This research was done in the context of the "Induction of Linguistic Knowledge" (ILK) research programme, supported partially by the Netherlands Organization for Scientific Research (NWO). The author wishes to thank Walter Daelemans, Jakub Zavrel, Ton Weijters, Ko van der Sloot, and the other members of the Tilburg ILK group for stimulating discussions and support. Software code for the RISE algorithm was kindly provided by Pedro Domingos; Dietrich Wettschereck kindly granted permission to use his implementation of NGE.

References

Abney, S. 1991. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Aha, D. W. 1992. Generalizing from case studies: a case study. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 1–10, San Mateo, CA. Morgan Kaufmann.

Aha, D. W. 1997. Lazy learning: Special issue editorial. *Artificial Intelligence Review*, 11:7–10.

Aha, D. W., D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Atkeson, C., A. Moore, and S. Schaal. 1997. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5):11–73.

Baayen, R. H., R. Piepenbrock, and H. van Rijn. 1993. *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA.

C. Blake, E. Keogh and C.J. Merz. 1998. UCI repository of machine learning databases.

Cardie, Claire. 1994. *Domain Specific Knowledge Acquisition for Conceptual Sentence Analysis*. Ph.D. thesis, University of Massachusets, Amherst, MA.

Cardie, Claire. 1996. Automatic feature set selection for case-based learning of linguistic knowledge. In *Proc. of Conference on Empirical Methods in NLP*. University of Pennsylvania.

Clark, P. and R. Boswell. 1991. Rule induction with CN2: Some recent improvements. In *Proceedings of the Sixth European Working Session on Learning*, pages 151–163. Berlin: Springer Verlag.

Clark, P. and T. Niblett. 1989. The CN2 rule induction algorithm. *Machine Learning*, 3:261–284.

Collins, M.J and J. Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proc. of Third Workshop on Very Large Corpora*, Cambridge.

Cost, S. and S. Salzberg. 1993. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

Cover, T. M. and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

Daelemans, W. and A. Van den Bosch. 1992a. Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers and A. Nijholt, editors, *Proc.*

of TWLT3: Connectionism and Natural Language Processing, pages 27–37, Enschede. Twente University.

Daelemans, W. and A. Van den Bosch. 1992b. A neural network for hyphenation. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks 2*, volume 2, pages 1647–1650, Amsterdam. North-Holland.

Daelemans, W., A. Van den Bosch, and A. Weijters. 1997. IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Daelemans, W., A. Van den Bosch, and J. Zavrel. 1998, forthcoming. Forgetting exceptions is harmful in language learning. *Machine Learning*.

Daelemans, Walter, Peter Berck, and Steven Gillis. 1997. Data mining as a method for linguistic analysis: Dutch diminutives. *Folia Linguistica*, XXXI(1-2).

Daelemans, Walter, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I.Dagan, editors, *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

Devijver, P. A. and J. Kittler. 1980. On the edited nearest neighbor rule. In *Proceedings of the Fifth International Conference on Pattern Recognition*. The Institute of Electrical and Electronics Engineers.

Devijver, P. .A. and J. Kittler. 1982. *Pattern recognition. A statistical approach*. Prentice-Hall, London, UK.

Dietterich, T. G. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7).

Dietterich, T. G., H. Hild, and G. Bakiri. 1995. A comparison of ID3 and backpropagation for English text-to-speech mapping. *Machine Learning*, 19(1):5–28.

Domingos, P. 1995. The rise 2.0 system: A case study in multistrategy learning. Technical Report 95-2, University of California at Irvine, Department of Information and Computer Science, Irvine, CA.

Domingos, P. 1996. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168.

Fix, E. and J. L. Hodges. 1951. Disciminatory analysis—nonparametric discrimination; consistency properties. Technical Report Project 21-49-004, Report No. 4, USAF School of Aviation Medicine.

Gates, G. W. 1972. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18:431–433.

Golding, A. R. and D. Roth. 1998, forthcoming. A Winnow-based approach to spelling correction. *Machine Learning*.

Hart, P. E. 1968. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516.

Kolodner, J. 1993. *Case-based reasoning*. Morgan Kaufmann, San Mateo, CA.

Kononenko, I. 1994. Estimating attributes: Analysis and extensions of relief. In *Proceedings of ECML'94*, pages 171–182.

Langley, P., W. Iba, and K. Thompson. 1992. An analysis of bayesian classifiers. In *Proceedings of the Tenth Annual Conference on Artificial Intelligence*, pages 223–228. AAAI Press and MIT Press.

Lehnert, W. 1987. Case-based problem solving with a large knowledge base of learned cases. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 301–306, Los Altos, CA. Morgan Kaufmann.

Marcus, M., B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Michalski, R. S. 1983. A theory and methodology of inductive learning. *Artificial Intelligence*, 11:111–161.

Niblett, T. 1987. Constructing decision trees in noisy domains. In *Proceedings of the Second European Working Session on Learning*, pages 67–78, Bled, Yugoslavia. Sigma.

Quinlan, J.R. 1986. Induction of Decision Trees. *Machine Learning*, 1:81–206.

Quinlan, J.R. 1993. C4.5*: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Ramshaw, L.A. and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of third workshop on very large corpora*, pages 82–94, June.

Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. In *Proc. of the Conference on Empirical Methods in Natural Language Processing, May 17-18, 1996, University of Pennsylvania*.

Ratnaparkhi, A. 1997. A linear observed time statistical parser based on maximum entropy models. Technical Report cmp-lg/9706014, Computation and Language, http://xxx.lanl.gov/list/cmp-lg/, June.

Ratnaparkhi, A., J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Workshop on Human Language Technology*, Plainsboro, NJ, March. ARPA.

Salzberg, S. 1991. A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309.

Salzberg, S. L. 1997. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3).

Sejnowski, T. J. and C. S. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

Shavlik, J. W. and T. G. Dietterich, editors. 1990. *Readings in Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Shavlik, J. W., R. J. Mooney, and G. G. Towell. 1991. An experimental comparison of symbolic and connectionist learning algorithms. *Machine Learning*, 6:111–143.

Stanfill, C. 1987. Memory-based reasoning applied to English pronunciation. In *Proceedings of the Sixth National Conference on Artificial Intelligence (*AAAI-87*)*, pages 577–581, Los Altos, CA. Morgan Kaufmann.

Stanfill, C. and D. Waltz. 1986. Toward memory-based reasoning. *Communications of the* ACM, 29(12):1213–1228, December.

Swonger, C. W. 1972. Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. In S. Watanabe, editor, *Frontiers of Pattern Recognition*. Academic Press, Orlando, Fla, pages 511–519.

Tomek, I. 1976. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(6):448–452.

Van den Bosch, A. 1997. *Learning to pronounce written words: A study in inductive language learning*. Ph.D. thesis, Universiteit Maastricht.

Van den Bosch, A., W. Daelemans, and A. Weijters. 1996. Morphological analysis as classification: an inductive-learning approach. In K. Oflazer and H. Somers, editors, *Proceedings of the Second International Conference on New Methods in Natural Language Processing, NeMLaP-2, Ankara, Turkey*, pages 79–89.

Vapnik, V. and L. Bottou. 1993. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 5(6):893–909.

Veenstra, J. B. 1998, forthcoming. Fast np chunking using memory-based learning techniques. In *Proceedings of BENELEARN'98*, Wageningen, The Netherlands.

Weijters, A. 1991. A simple look-up procedure superior to nettalk? In *Proceedings of the International Conference on Artificial Neural Networks* - ICANN-*91, Espoo, Finland*.

Weiss, S. and C. Kulikowski. 1991. *Computer systems that learn*. San Mateo, CA: Morgan Kaufmann.

Wettschereck, D., D. W. Aha, and T. Mohri. 1997. A review and comparative evaluation of feature-weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review, special issue on Lazy Learning*, 11:273–314.

Wettschereck, D. and T. G. Dietterich. 1995. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:1–25.

Wilson, D. 1972. Asymptotic properties of nearest neighbor rules using edited data. *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics*, 2:408–421.

Wolpert, D. 1989. Constructing a generalizer superior to NETtalk via mathematical theory of generalization. *Neural Networks*, 3:445–452.

Zavrel, J., W. Daelemans, and J. Veenstra. 1997. Resolving pp attachment ambiguities with memory-based learning. In M. Ellison, editor, *Proc. of the Workshop on Computational Language Learning (CoNLL'97)*, ACL, Madrid.